Reference Manual

Generated by Doxygen 1.6.1

Mon Aug 28 14:15:38 2017

Contents

1 Main Page

API for SLS detectors data acquisition

Although the SLS detectors group delvelops several types of detectors (1/2D, counting/integrating etc.) it is common interest of the group to use a common platfor for data acquisition

The architecture of the acquisitions system is intended as follows:

- A socket server running on the detector (or more than one in some special cases)
- C++ classes common to all detectors for client-server communication. These can be supplied to users as libraries and embedded also in acquisition systems which are not developed by the SLS
- the possibility of using a Qt-based graphical user interface (with eventually root analisys capabilities)
- the possibility of running all commands from command line. In order to ensure a fast operation of this so called "text client" the detector parameters should not be re-initialized everytime. For this reason a shared memory block is allocated where the main detector flags and parameters are stored
- a Root library for data postprocessing and detector calibration (energy, angle).

slsDetectorUsers is a class to control the detector which should be instantiated by the users in their acquisition software (EPICS, spec etc.). A callback for dislaying the data can be registered. More advanced configuration functions are not implemented and can be written in a configuration file tha can be read/written.

slsReceiverUsers is a class to receive the data for detectors with external data receiver (e.g. GOTTHARD). Callbacks can be registered to process the data or save them in specific formats.

detectorData is a structure containing the data and additional information which is used to return the data e.g. to the GUI for displaying them.

You can find examples of how this classes can be instatiated in mainClient.cpp and mainReceiver.cpp

Authors:

Anna Bergamaschi, Dhanya Thattil

Version:

3.0

Currently supported detectors

- MYTHEN
- GOTTHARD controls

- GOTTHARD data receiver
- EIGER
- JUNGFRAU

2 Class Documentation

2.1 detectorData Class Reference

data structure to hold the detector data after postprocessing (e.g. to plot, store in a root tree etc.)

`#include <detectorData.h>`

Public Member Functions

- detectorData (double *val=NULL, double *err=NULL, double *ang=NULL, double p_ind=-1, const char *fname="", int np=-1, int ny=1)
  The constructor.
- ~detectorData ()
  The destructor deletes also the arrays pointing to data/errors/angles if not NULL.

Public Attributes

- double * values
  pointer to the data
- double * errors
  pointer to the errors
- double * angles
  pointer to the angles (NULL if no angular conversion)
- double progressIndex
  file index
- char fileName [1000]
  file name
- int npoints
  number of points
- int npy
  dimensions in y coordinate

2.1.1 Detailed Description

data structure to hold the detector data after postprocessing (e.g. to plot, store in a root tree etc.)

Definition at line 9 of file detectorData.h.

2.1.2 Constructor & Destructor Documentation

2.1.2.1 detectorData::detectorData (double * *val* = NULL, double * *err* = NULL, double * *ang* = NULL, double *p_ind* = -1, const char * *fname* = "", int *np* = -1, int *ny* = 1) [inline]

The constructor.

Parameters:

val pointer to the data

err pointer to errors

ang pointer to the angles

f_ind file index

fname file name to which the data are saved

np number of points in x coordinate defaults to the number of detector channels (1D detector)

ny dimension in y (1D detector)

Definition at line 20 of file detectorData.h.

2.1.2.2 detectorData::~detectorData () [inline]

The destructor deletes also the arrays pointing to data/errors/angles if not NULL.

Definition at line 27 of file detectorData.h.

2.1.3 Member Data Documentation

2.1.3.1 double* detectorData::angles

pointer to the angles (NULL if no angular conversion)

Definition at line 31 of file detectorData.h.

2.1.3.2 double* detectorData::errors

pointer to the errors

Definition at line 30 of file detectorData.h.

2.1.3.3 char detectorData::fileName[1000]

file name

Definition at line 33 of file detectorData.h.

2.1.3.4 int detectorData::npoints

number of points

Definition at line 34 of file detectorData.h.

2.1.3.5 int detectorData::npy

dimensions in y coordinate

Definition at line 35 of file detectorData.h.

2.1.3.6 double detectorData::progressIndex

file index

Definition at line 32 of file detectorData.h.

2.1.3.7 double* detectorData::values

pointer to the data

Definition at line 27 of file detectorData.h.

The documentation for this class was generated from the following file:

- detectorData.h

2.2 slsDetectorUsers Class Reference

Class for detector functionalitiesto embed the detector controls in the users custom interface e.g. EPICS, Lima etc.

`#include <slsDetectorUsers.h>`

Public Member Functions

- slsDetectorUsers (int id=0)
  default constructor
- virtual ~slsDetectorUsers ()
  virtual destructor
- string getDetectorDeveloper ()
  useful to define subset of working functions
- int setOnline (int const online=-1)
  sets the onlineFlag
- void startMeasurement ()
  start measurement and acquires
- int stopMeasurement ()
  stop measurement
- int getDetectorStatus ()
  get run status
- string getFilePath ()
  returns the default output files path
- string setFilePath (string s)
  sets the default output files path
- string getFileName ()
- string setFileName (string s)
  sets the default output files path
- int getFileIndex ()
- int setFileIndex (int i)
  sets the default output file index
- string getFlatFieldCorrectionDir ()
  get flat field corrections file directory

- string setFlatFieldCorrectionDir (string dir)
  set flat field corrections file directory
- string getFlatFieldCorrectionFile ()
  get flat field corrections file name
- int setFlatFieldCorrectionFile (string fname="")
  set flat field correction file
- int enableFlatFieldCorrection (int i=-1)
  enable/disable flat field corrections (without changing file name)
- int enableCountRateCorrection (int i=-1)
  enable/disable count rate corrections
- int enablePixelMaskCorrection (int i=-1)
  enable/disable bad channel corrections
- int enableAngularConversion (int i=-1)
  enable/disable angular conversion
- int enableWriteToFile (int i=-1)
- int setPositions (int nPos, double *pos)
  set positions for the acquisition
- int getPositions (double *pos=NULL)
  get positions for the acquisition
- int setDetectorSize (int x0=-1, int y0=-1, int nx=-1, int ny=-1)
  sets the detector size
- int getDetectorSize (int &x0, int &y0, int &nx, int &ny)
  gets detector size
- int getMaximumDetectorSize (int &nx, int &ny)
  setsthe maximum detector size
- int setBitDepth (int i=-1)
  set/get dynamic range
- int setSettings (int isettings=-1)
  set detector settings
- int getThresholdEnergy ()
  get threshold energy

- int setThresholdEnergy (int e_eV)

  set threshold energy

- double setExposureTime (double t=-1, bool inseconds=false)

  set/get exposure time value

- double setExposurePeriod (double t=-1, bool inseconds=false)

  set/get exposure period

- double setDelayAfterTrigger (double t=-1, bool inseconds=false)

  set/get delay after trigger

- int64_t setNumberOfGates (int64_t t=-1)

  set/get number of gates

- int64_t setNumberOfFrames (int64_t t=-1)

  set/get number of frames i.e. number of exposure per trigger

- int64_t setNumberOfCycles (int64_t t=-1)

  set/get number of cycles i.e. number of triggers

- int setTimingMode (int pol=-1)

  set/get the external communication mode

- int readConfigurationFile (string const fname)

  Reads the configuration file -- will contain all the informations needed for the configuration (e.g. for a PSI detector caldir, settingsdir, angconv, badchannels, hostname etc.).

- int dumpDetectorSetup (string const fname)

  Reads the parameters from the detector and writes them to file.

- int retrieveDetectorSetup (string const fname)

  Loads the detector setup from file.

- string getDetectorType ()

  useful for data plotting etc.

- int setReceiverMode (int n=-1)

  sets the mode by which gui requests data from receiver

- void registerDataCallback (int(∗userCallback)(detectorData ∗d, int f, int s, void ∗), void ∗pArg)

  register calbback for accessing detector final data

- void registerRawDataCallback (int(∗userCallback)(double ∗p, int n, void ∗), void ∗pArg)

register callback for accessing raw data - if the rawDataCallback is registered, no filewriting/postprocessing will be carried on automatically by the software - the raw data are deleted by the software

- virtual void initDataset (int refresh)

  function to initalize a set of measurements (reset binning if angular conversion, reset summing otherwise) - can be overcome by the user's functions thanks to the virtual property

- virtual void addFrame (double *data, double pos, double i0, double t, string fname, double var)

  adds frame to merging/summation - can be overcome by the user's functions thanks to the virtual property

- virtual void finalizeDataset (double *a, double *v, double *e, int &np)

  finalizes the data set returning the array of angles, values and errors to be used as final data - can be overcome by the user's functions thanks to the virtual property

- int enableDataStreamingFromReceiver (int i=-1)
- int64_t getModuleFirmwareVersion ()
- int64_t getModuleSerialNumber (int imod=-1)
- int64_t getDetectorFirmwareVersion ()
- int64_t getDetectorSerialNumber ()
- int64_t getDetectorSoftwareVersion ()
- int64_t getThisSoftwareVersion ()
- void registerAcquisitionFinishedCallback (int(*func)(double, int, void *), void *pArg)

  register calbback for accessing detector final data

- void registerGetPositionCallback (double(*func)(void *), void *arg)

  register calbback for reading detector position

- void registerConnectChannelsCallback (int(*func)(void *), void *arg)

  register callback for connecting to the epics channels

- void registerDisconnectChannelsCallback (int(*func)(void *), void *arg)

  register callback to disconnect the epics channels

- void registerGoToPositionCallback (int(*func)(double, void *), void *arg)

  register callback for moving the detector

- void registerGoToPositionNoWaitCallback (int(*func)(double, void *), void *arg)

  register callback for moving the detector without waiting

- void registerGetI0Callback (double(*func)(int, void *), void *arg)

  register calbback reading to I0

- string putCommand (int narg, char *args[ ], int pos=-1)

  sets parameters in command interface http://www.psi.ch/detectors/UsersSupportEN/slsDetectorC

- string getCommand (int narg, char *args[ ], int pos=-1)

  gets parameters in command interface http://www.psi.ch/detectors/UsersSupportEN/slsDetectorC

Static Public Member Functions

- static string runStatusType (int s)

  returns string from run status index

- static int getDetectorSettings (string s)

  returns detector settings string from index

- static string getDetectorSettings (int s)

  returns detector settings string from index

- static string getTimingMode (int f)

  returns external communication mode string from index

- static int getTimingMode (string s)

  returns external communication mode string from index

2.2.1 Detailed Description

Class for detector functionalitiesto embed the detector controls in the users custom interface e.g. EPICS, Lima etc. The slsDetectorUsers class is a minimal interface class which should be instantiated by the users in their acquisition software (EPICS, spec etc.). More advanced configuration functions are not implemented and can be written in a configuration or parameters file that can be read/written.

Definition at line 85 of file slsDetectorUsers.h.

2.2.2 Constructor & Destructor Documentation

2.2.2.1 slsDetectorUsers::slsDetectorUsers (int *id* = 0)

default constructor

2.2.2.2 virtual slsDetectorUsers::~slsDetectorUsers () [virtual]

virtual destructor

2.2.3 Member Function Documentation

2.2.3.1 virtual void slsDetectorUsers::addFrame (double * *data*, double *pos*, double *i0*, double *t*, string *fname*, double *var*) [virtual]

adds frame to merging/summation - can be overcome by the user's functions thanks to the virtual property

Parameters:

data pointer to the raw data

pos encoder position

i0 beam monitor readout for intensity normalization (if 0 not performed)

t exposure time in seconds, required only if rate corrections

fname file name (unused since filewriting would be performed by the user)

var optional parameter - unused.

2.2.3.2 int slsDetectorUsers::dumpDetectorSetup (string const *fname*)

Reads the parameters from the detector and writes them to file.

Parameters:

fname file to write to

Returns:

OK or FAIL

2.2.3.3 int slsDetectorUsers::enableAngularConversion (int *i* = -1)

enable/disable angular conversion

Parameters:

i 0 disables, 1 enables, -1 gets

Returns:

0 if angular conversion disabled, 1 if enabled

2.2.3.4 int slsDetectorUsers::enableCountRateCorrection (int *i* = -1)

enable/disable count rate corrections

Parameters:

i 0 disables, 1 enable, -1 gets

Returns:

0 if count corrections disabled, 1 if enabled

2.2.3.5 int slsDetectorUsers::enableDataStreamingFromReceiver (int *i* = -1)

Enable data streaming from receiver (zmq)

Parameters:

i 1 to set, 0 to reset and -1 to get

Returns:

data streaming enable

2.2.3.6 int slsDetectorUsers::enableFlatFieldCorrection (int *i* = -1)

enable/disable flat field corrections (without changing file name)

Parameters:

i 0 disables, 1 enables, -1 gets

Returns:

0 if ff corrections disabled, 1 if enabled

2.2.3.7 int slsDetectorUsers::enablePixelMaskCorrection (int *i* = -1)

enable/disable bad channel corrections

Parameters:

i 0 disables, 1 enables, -1 gets

Returns:

0 if bad channels corrections disabled, 1 if enabled

2.2.3.8 int slsDetectorUsers::enableWriteToFile (int *i* = -1)

Enable write file function included

2.2.3.9 virtual void slsDetectorUsers::finalizeDataset (double * *a*, double * *v*, double * *e*, int & *np*) [virtual]

finalizes the data set returning the array of angles, values and errors to be used as final data - can be overcome by the user's functions thanks to the virtual property

Parameters:

a pointer to the array of angles - can be null if no angular coversion is required

v pointer to the array of values

e pointer to the array of errors

np reference returning the number of points

2.2.3.10 string slsDetectorUsers::getCommand (int *narg*, char * *args*[ ], int *pos* = -1)

gets parameters in command interface http://www.psi.ch/detectors/UsersSupportEN/slsDetecto

Parameters:

narg value to be set

args value to be set

pos position of detector in multislsdetector list

Returns:

answer string

2.2.3.11 string slsDetectorUsers::getDetectorDeveloper ()

useful to define subset of working functions

Returns:

"PSI" or "Dectris"

2.2.3.12 int64_t slsDetectorUsers::getDetectorFirmwareVersion ()

get get Detector Firmware Version

Returns:

id

2.2.3.13 int64_t slsDetectorUsers::getDetectorSerialNumber ()

get get Detector Serial Number

Returns:

id

2.2.3.14 static string slsDetectorUsers::getDetectorSettings (int *s*) [inline, static]

returns detector settings string from index

Parameters:

s settings index

Returns:

standard, fast, highgain, dynamicgain, lowgain, mediumgain, veryhighgain, undefined when wrong index

Definition at line 586 of file slsDetectorUsers.h.

2.2.3.15 static int slsDetectorUsers::getDetectorSettings (string *s*) [inline, static]

returns detector settings string from index

Parameters:

s can be standard, fast, highgain, dynamicgain, lowgain, mediumgain, veryhighgain

Returns:

setting index (-1 unknown string)

Definition at line 572 of file slsDetectorUsers.h.

2.2.3.16 int slsDetectorUsers::getDetectorSize (int & *x0*, int & *y0*, int & *nx*, int & *ny*)

gets detector size

Parameters:

x0 horizontal position origin in channel number

y0 vertical position origin in channel number

nx number of channels in horiziontal

ny number of channels in vertical

Returns:

OK/FAIL

2.2.3.17 int64_t slsDetectorUsers::getDetectorSoftwareVersion ()

get get Detector Software Version

Returns:

id

2.2.3.18 int slsDetectorUsers::getDetectorStatus ()

get run status

Returns:

status mask

2.2.3.19 string slsDetectorUsers::getDetectorType ()

useful for data plotting etc.

Returns:

Mythen, Eiger, Gotthard etc.

2.2.3.20 int slsDetectorUsers::getFileIndex ()

Returns:

the default output file index

2.2.3.21 string slsDetectorUsers::getFileName ()

Returns:

the default output files root name

2.2.3.22 string slsDetectorUsers::getFilePath ()

returns the default output files path

2.2.3.23 string slsDetectorUsers::getFlatFieldCorrectionDir ()

get flat field corrections file directory

Returns:

flat field correction file directory

2.2.3.24 string slsDetectorUsers::getFlatFieldCorrectionFile ()

get flat field corrections file name

Returns:

flat field correction file name

2.2.3.25 int slsDetectorUsers::getMaximumDetectorSize (int & *nx*, int & *ny*)

setsthe maximum detector size

Parameters:

x0 horizontal position origin in channel number

y0 vertical position origin in channel number

nx number of channels in horiziontal

ny number of channels in vertical

Returns:

OK/FAIL

2.2.3.26 int64_t slsDetectorUsers::getModuleFirmwareVersion ()

get get Module Firmware Version

Returns:

id

2.2.3.27 int64_t slsDetectorUsers::getModuleSerialNumber (int *imod* = `-1`)

get get Module Serial Number

Parameters:

imod module number

Returns:

id

2.2.3.28 int slsDetectorUsers::getPositions (double * *pos* = `NULL`)

get positions for the acquisition

Parameters:

pos array which will contain the encoder positions

Returns:

number of positions

2.2.3.29 int64_t slsDetectorUsers::getThisSoftwareVersion ()

get this Software Version

Returns:

id

2.2.3.30 int slsDetectorUsers::getThresholdEnergy ()

get threshold energy

Returns:

current threshold value for imod in ev (-1 failed)

2.2.3.31 static int slsDetectorUsers::getTimingMode (string *s*) [inline, static]

returns external communication mode string from index

Parameters:

f index for communication mode

Returns:

auto, trigger, ro_trigger, gating, triggered_gating, unknown when wrong mode

Definition at line 622 of file slsDetectorUsers.h.

2.2.3.32 static string slsDetectorUsers::getTimingMode (int *f*) [inline, static]

returns external communication mode string from index

Parameters:

f index for communication mode

Returns:

auto, trigger, ro_trigger, gating, triggered_gating, unknown when wrong mode

Definition at line 606 of file slsDetectorUsers.h.

2.2.3.33 virtual void slsDetectorUsers::initDataset (int *refresh*) [virtual]

function to initalize a set of measurements (reset binning if angular conversion, reset summing otherwise) - can be overcome by the user's functions thanks to the virtual property

Parameters:

refresh if 1, all parameters like ffcoefficients, badchannels, ratecorrections etc. are reset (should be called at least onece with this option), if 0 simply reset merging/ summation

2.2.3.34 string slsDetectorUsers::putCommand (int *narg*, char * *args*[ ], int *pos* = -1)

sets parameters in command interface http://www.psi.ch/detectors/UsersSupportEN/slsDetector

Parameters:

narg value to be set

args value to be set

pos position of detector in multislsdetector list

Returns:

answer string

2.2.3.35 int slsDetectorUsers::readConfigurationFile (string const *fname*)

Reads the configuration file -- will contain all the informations needed for the configuration (e.g. for a PSI detector caldir, settingsdir, angconv, badchannels, hostname etc.).

Parameters:

fname file name

Returns:

OK or FAIL

2.2.3.36 void slsDetectorUsers::registerAcquisitionFinishedCallback (int(∗)(double, int, void ∗) *func*, void ∗ *pArg*)

register calbback for accessing detector final data

Parameters:

func function to be called at the end of the acquisition. gets detector status and progress index as arguments

2.2.3.37 void slsDetectorUsers::registerConnectChannelsCallback (int(∗)(void ∗) *func*, void ∗ *arg*)

register callback for connecting to the epics channels

Parameters:

func function for connecting to the epics channels

2.2.3.38 void slsDetectorUsers::registerDataCallback (int(∗)(detectorData ∗d, int f, int s, void ∗) *userCallback*, void ∗ *pArg*)

register calbback for accessing detector final data

Parameters:

userCallback function for plotting/analyzing the data. Its arguments are the data structure d and the frame number f, s is for subframe number for eiger for 32 bit mode

2.2.3.39 void slsDetectorUsers::registerDisconnectChannelsCallback (int(∗)(void ∗) *func*, void ∗ *arg*)

register callback to disconnect the epics channels

Parameters:

func function to disconnect the epics channels

2.2.3.40 void slsDetectorUsers::registerGetI0Callback (double(∗)(int, void ∗) *func*, void ∗ *arg*)

register calbback reading to I0

Parameters:

func function for reading the I0 (called with parameter 0 before the acquisition, 1 after and the return value used as I0)

2.2.3.41 void slsDetectorUsers::registerGetPositionCallback (double(∗)(void ∗) *func*, void ∗ *arg*)

register calbback for reading detector position

Parameters:

func function for reading the detector position

2.2.3.42 void slsDetectorUsers::registerGoToPositionCallback (int(∗)(double, void ∗) *func*, void ∗ *arg*)

register callback for moving the detector

Parameters:

func function for moving the detector

2.2.3.43 void slsDetectorUsers::registerGoToPositionNoWaitCallback (int(∗)(double, void ∗) *func*, void ∗ *arg*)

register callback for moving the detector without waiting

Parameters:

func function for moving the detector

2.2.3.44 void slsDetectorUsers::registerRawDataCallback (int(∗)(double ∗p, int n, void ∗) *userCallback*, void ∗ *pArg*)

register callback for accessing raw data - if the rawDataCallback is registered, no filewriting/postprocessing will be carried on automatically by the software - the raw data are deleted by the software

Parameters:

userCallback function for postprocessing and saving the data - p is the pointer to the data, n is the number of channels

2.2.3.45 int slsDetectorUsers::retrieveDetectorSetup (string const *fname*)

Loads the detector setup from file.

Parameters:

fname file to read from

Returns:

OK or FAIL

2.2.3.46 static string slsDetectorUsers::runStatusType (int *s*) [inline, static]

returns string from run status index

Parameters:

s run status index

Returns:

string error, waiting, running, data, finished or unknown when wrong index

Definition at line 554 of file slsDetectorUsers.h.

2.2.3.47 int slsDetectorUsers::setBitDepth (int *i* = -1)

set/get dynamic range

Parameters:

i dynamic range (-1 get)

Returns:

current dynamic range

2.2.3.48 double slsDetectorUsers::setDelayAfterTrigger (double *t* = -1, bool *inseconds* = false)

set/get delay after trigger

Parameters:

t time in ns (-1 gets)

inseconds true if the value is in s, else ns

Returns:

timer set value in ns, or s if specified

2.2.3.49 int slsDetectorUsers::setDetectorSize (int *x0* = -1, int *y0* = -1, int *nx* = -1, int *ny* = -1)

sets the detector size

Parameters:

x0 horizontal position origin in channel number (-1 unchanged)

y0 vertical position origin in channel number (-1 unchanged)

nx number of channels in horiziontal (-1 unchanged)

ny number of channels in vertical (-1 unchanged)

Returns:

OK/FAIL

2.2.3.50 double slsDetectorUsers::setExposurePeriod (double *t* = -1, bool *inseconds* = false)

set/get exposure period

Parameters:

t time in ns (-1 gets)

inseconds true if the value is in s, else ns

Returns:

timer set value in ns, or s if specified

2.2.3.51 double slsDetectorUsers::setExposureTime (double *t* = `-1`, bool *inseconds* = `false`)

set/get exposure time value

Parameters:

t time in sn (-1 gets)

inseconds true if the value is in s, else ns

Returns:

timer set value in ns, or s if specified

2.2.3.52 int slsDetectorUsers::setFileIndex (int *i*)

sets the default output file index

Parameters:

i file index

Returns:

the default output file index

2.2.3.53 string slsDetectorUsers::setFileName (string *s*)

sets the default output files path

Parameters:

s file name

Returns:

the default output files root name

2.2.3.54 string slsDetectorUsers::setFilePath (string *s*)

sets the default output files path

Parameters:

s file path

Returns:

file path

2.2.3.55 string slsDetectorUsers::setFlatFieldCorrectionDir (string *dir*)

set flat field corrections file directory

Parameters:

dir flat field correction file directory

Returns:

flat field correction file directory

2.2.3.56 int slsDetectorUsers::setFlatFieldCorrectionFile (string *fname* = "")

set flat field correction file

Parameters:

fname name of the flat field file (or "" if disable)

Returns:

0 if disable (or file could not be read), >0 otherwise

2.2.3.57 int64_t slsDetectorUsers::setNumberOfCycles (int64_t *t* = -1)

set/get number of cycles i.e. number of triggers

Parameters:

t number of frames (-1 gets)

Returns:

number of frames

2.2.3.58 int64_t slsDetectorUsers::setNumberOfFrames (int64_t *t* = -1)

set/get number of frames i.e. number of exposure per trigger

Parameters:

t number of frames (-1 gets)

Returns:

number of frames

2.2.3.59 int64_t slsDetectorUsers::setNumberOfGates (int64_t *t* = -1)

set/get number of gates

Parameters:

t number of gates (-1 gets)

Returns:

number of gates

2.2.3.60 int slsDetectorUsers::setOnline (int const *online* = -1)

sets the onlineFlag

Parameters:

online can be: -1 returns wether the detector is in online (1) or offline (0) state; 0 detector in offline state; 1 detector in online state

Returns:

0 (offline) or 1 (online)

2.2.3.61 int slsDetectorUsers::setPositions (int *nPos*, double * *pos*)

set positions for the acquisition

Parameters:

nPos number of positions

pos array with the encoder positions

Returns:

number of positions

2.2.3.62 int slsDetectorUsers::setReceiverMode (int *n* = −1)

sets the mode by which gui requests data from receiver

Parameters:

n is 0 for random requests for fast acquisitions and greater than 0 for nth read requests

Returns:

the mode set in the receiver

2.2.3.63 int slsDetectorUsers::setSettings (int *isettings* = −1)

set detector settings

Parameters:

isettings settings index (-1 gets)

Returns:

current settings

2.2.3.64 int slsDetectorUsers::setThresholdEnergy (int *e_eV*)

set threshold energy

Parameters:

e_eV threshold in eV

Returns:

current threshold value for imod in ev (-1 failed)

2.2.3.65 int slsDetectorUsers::setTimingMode (int *pol* = -1)

set/get the external communication mode

Parameters:

pol value to be set

See also:

getTimingMode

Returns:

current external communication mode

2.2.3.66 void slsDetectorUsers::startMeasurement ()

start measurement and acquires

Returns:

OK/FAIL

2.2.3.67 int slsDetectorUsers::stopMeasurement ()

stop measurement

Returns:

OK/FAIL

The documentation for this class was generated from the following file:

- slsDetectorUsers.h

2.3 slsReceiverUsers Class Reference

Class for implementing the SLS data receiver in the users application. Callbacks can be defined for processing and/or saving data.

`#include <slsReceiverUsers.h>`

Public Member Functions

- slsReceiverUsers (int argc, char ∗argv[ ], int &success)
- ∼slsReceiverUsers ()
- int start ()
- void stop ()
- int64_t getReceiverVersion ()
- void registerCallBackStartAcquisition (int(∗func)(char ∗filepath, char ∗filename, uint64_t fileindex, uint32_t datasize, void ∗), void ∗arg)
- void registerCallBackAcquisitionFinished (void(∗func)(uint64_t nf, void ∗), void ∗arg)
- void registerCallBackRawDataReady (void(∗func)(uint64_t frameNumber, uint32_t expLength, uint32_t packetNumber, uint64_t bunchId, uint64_t timestamp, uint16_t modId, uint16_t xCoord, uint16_t yCoord, uint16_t zCoord, uint32_t debug, uint16_t roundRNumber, uint8_t detType, uint8_t version, char ∗datapointer, uint32_t datasize, void ∗), void ∗arg)

Public Attributes

- slsReceiver ∗ receiver

2.3.1 Detailed Description

Class for implementing the SLS data receiver in the users application. Callbacks can be defined for processing and/or saving data. slsReceiverUsers is a class that can be instantiated in the users software to receive the data from the detectors. Callbacks can be defined for processing and/or saving data

Definition at line 16 of file slsReceiverUsers.h.

2.3.2 Constructor & Destructor Documentation

2.3.2.1 slsReceiverUsers::slsReceiverUsers (int *argc*, char ∗ *argv*[ ], int & *success*)

Constructor reads config file, creates socket, assigns function table

Parameters:

argc from command line

argv from command line

succecc socket creation was successfull

2.3.2.2 slsReceiverUsers::∼slsReceiverUsers ()

Destructor

2.3.3 Member Function Documentation

2.3.3.1 int64_t slsReceiverUsers::getReceiverVersion ()

get get Receiver Version

Returns:

id

2.3.3.2 void slsReceiverUsers::registerCallBackAcquisitionFinished (void(∗)(uint64_t nf, void ∗) *func*, void ∗ *arg*)

register callback for end of acquisition

Parameters:

func end of acquisition callback. Argument nf is total frames caught

Returns:

nothing

2.3.3.3 void slsReceiverUsers::registerCallBackRawDataReady (void(∗)(uint64_t frameNumber, uint32_t expLength, uint32_t packetNumber, uint64_t bunchId, uint64_t timestamp, uint16_t modId, uint16_t xCoord, uint16_t yCoord, uint16_t zCoord, uint32_t debug, uint16_t roundRNumber, uint8_t detType, uint8_t version, char ∗datapointer, uint32_t datasize, void ∗) *func*, void ∗ *arg*)

register callback to be called when data are available (to process and/or save the data).

Parameters:

func raw data ready callback. arguments are frameNumber, expLength, packetNumber, bunchId, timestamp, modId, xCoord, yCoord, zCoord, debug, roundRNumber, detType, version, dataPointer, dataSize

Returns:

nothing

2.3.3.4 void slsReceiverUsers::registerCallBackStartAcquisition (int(∗)(char ∗filepath, char ∗filename, uint64_t fileindex, uint32_t datasize, void ∗) *func*, void ∗ *arg*)

register calbback for starting the acquisition

Parameters:

func callback to be called when starting the acquisition. Its arguments are filepath, filename, fileindex, datasize

Returns:

value is insignificant at the moment, we write depending on file write enable, users get data to write depending on call backs registered

2.3.3.5 int slsReceiverUsers::start ()

starts listening on the TCP port for client comminication

Returns:

0 for success or 1 for FAIL in creating TCP server

2.3.3.6 void slsReceiverUsers::stop ()

stops listening to the TCP & UDP port and exit receiver program

2.3.4 Member Data Documentation

2.3.4.1 slsReceiver* slsReceiverUsers::receiver

Definition at line 75 of file slsReceiverUsers.h.

The documentation for this class was generated from the following file:

- slsReceiverUsers.h

3 File Documentation

3.1 detectorData.h File Reference

```
#include <unistd.h>
#include <cstring>
```

Classes

- class detectorData

  data structure to hold the detector data after postprocessing (e.g. to plot, store in a root tree etc.)

3.2 mainClient.cpp File Reference

```
#include "slsDetectorUsers.h"
#include "detectorData.h"
#include <iostream>
#include <cstdlib>
```

Functions

- int dataCallback (detectorData *pData, int iframe, int isubframe, void *pArg)
- int main (int argc, char **argv)

3.2.1 Detailed Description

This file is an example of how to implement the slsDetectorUsers class You can compile it linking it to the slsDetector library

g++ mainClient.cpp -L lib -lSlsDetector -L/usr/lib64/ -L lib2 -lzmq -pthread -lrt -lm -lstdc++

where,

lib is the location of libSlsDetector.so

lib2 is the location of the libzmq.a. [libzmq.a is required only when using data call backs and enabling data streaming from receiver to client. It is linked in manual/manual-api from slsReceiverSoftware/include]

Definition in file mainClient.cpp.

3.2.2 Function Documentation

3.2.2.1 int dataCallback (detectorData * *pData*, int *iframe*, int *isubframe*, void * *pArg*)

Data Call back function defined

Parameters:

pData pointer to data structure received from the call back

iframe frame number of data passed

isubframe sub frame number of data passed (only valid for EIGER in 32 bit mode)

pArg pointer to object

Returns:

integer that is currently ignored

Definition at line 32 of file mainClient.cpp.

3.2.2.2 int main (int *argc*, char ** *argv*)

Example of a main program using the slsDetectorUsers class

- Arguments are optional
  - argv[1] : Configuration File
  - argv[2] : Measurement Setup File
  - argv[3] : Detector Id (default is zero)
- if specified, set ID from argv[3]
- slsDetectorUsers Object is instantiated with appropriate ID
- if specified, load configuration file (necessary at least the first time it is called to properly configure advanced settings in the shared memory)
- registering data callback
- if receiver exists, enable data streaming from receiver to get the data
- ensuring detector status is idle before starting acquisition. exiting if not idle
- if provided, load detector settings
- start measurement
- returning when acquisition is finished or data are avilable
- delete slsDetectorUsers object

Definition at line 46 of file mainClient.cpp.

3.3 mainReceiver.cpp File Reference

```
#include "sls_receiver_defs.h"
#include "slsReceiverUsers.h"
#include <iostream>
#include <string.h>
#include <signal.h>
#include <cstdlib>
#include <sys/types.h>
#include <sys/wait.h>
#include <string>
```

Defines

- #define NUM_RECEIVERS 1
- #define START_TCP_PORT 1954
- #define PRINT_IN_COLOR(c, f,...) printf ("\033[%dm" f RESET, 30 + c+1, ##__VA_ARGS__)

Functions

- void sigChildExitedHandler (int sig)
- void sigInterruptHandler (int p)
- int StartAcq (char *filepath, char *filename, uint64_t fileindex, uint32_t datasize, void *p)
- void AcquisitionFinished (uint64_t frames, void *p)
- void GetData (uint64_t frameNumber, uint32_t expLength, uint32_t packetNumber, uint64_t bunchId, uint64_t timestamp, uint16_t modId, uint16_t xCoord, uint16_t yCoord, uint16_t zCoord, uint32_t debug, uint16_t roundRNumber, uint8_t detType, uint8_t version, char *datapointer, uint32_t datasize, void *p)
- int main (int argc, char *argv[ ])

Variables

- pid_t childPid [NUM_RECEIVERS]
- bool keeprunning
- int numrunning

3.3.1 Detailed Description

This file is an example of how to implement the slsReceiverUsers class You can compile it linking it to the slsReceiver library

g++ mainReceiver.cpp -L lib -lSlsReceiver -L/usr/lib64/ -L lib2 -lzmq -pthread -lrt -lm -lstdc++

where,

lib is the location of lSlsReceiver.so

lib2 is the location of the libzmq.a. [libzmq.a is required only when using data call backs and enabling data streaming from receiver to client. It is linked in manual/manual-api from slsReceiverSoftware/include]

Definition in file mainReceiver.cpp.

3.3.2 Define Documentation

3.3.2.1 #define NUM_RECEIVERS 1

Definition at line 34 of file mainReceiver.cpp.

3.3.2.2 #define PRINT_IN_COLOR(c, f, ...) printf ("\033[%dm" f RESET, 30 + c+1, ##__VA_ARGS__)

Define Colors to print data call back in different colors for different recievers

Definition at line 38 of file mainReceiver.cpp.

3.3.2.3 #define START_TCP_PORT 1954

Define TCP Port of First Receiver, others are incremented by 1

Definition at line 36 of file mainReceiver.cpp.

3.3.3 Function Documentation

3.3.3.1 void AcquisitionFinished (uint64_t *frames*, void * *p*)

Acquisition Finished Call back

Parameters:

frames Number of frames caught

p pointer to object

Definition at line 93 of file mainReceiver.cpp.

3.3.3.2 void GetData (uint64_t *frameNumber*, uint32_t *expLength*, uint32_t *packetNumber*, uint64_t *bunchId*, uint64_t *timestamp*, uint16_t *modId*, uint16_t *xCoord*, uint16_t *yCoord*, uint16_t *zCoord*, uint32_t *debug*, uint16_t *roundRNumber*, uint8_t *detType*, uint8_t *version*, char * *datapointer*, uint32_t *datasize*, void * *p*)

Get Receiver Data Call back Prints in different colors(for each receiver process) the different headers for each image call back.

Parameters:

frameNumber frame number

expLength real time exposure length (in 100ns) or sub frame number (Eiger 32 bit mode only)

packetNumber number of packets caught for this frame

bunchId bunch id from beamline

timestamp time stamp in 10MHz clock (not implemented for most)

modId module id (not implemented for most)

xCoord x coordinates (detector id in 1D)

yCoord y coordinates (not implemented)

zCoord z coordinates (not implemented)

debug debug values if any

roundRNumber (not implemented)

detType detector type see :: detectorType

version version of standard header (structure format)

datapointer pointer to data

datasize data size in bytes

p pointer to object

Definition at line 117 of file mainReceiver.cpp.

3.3.3.3 int main (int *argc*, char * *argv*[ ])

Example of main program using the slsReceiverUsers class

- Defines in file for:
  - Default Number of receivers is 1
  - Default Start TCP port is 1954
- set default values: child process pid values to -1, keeprunning to true, numrunning to 0
- Catch signal SIGINT to close files and call destructors properly
- wait for all the SIGCHILD signals and decrease numrunningeach time a child process exits
- loop over number of receivers
- fork process to create child process
- if fork failed, raise SIGINT and kill all receiver objects
- if child process
- create slsReceiverUsers object with appropriate arguments (START_TCP_PORT incrementing by 1
- register callbacks. remember to set file write enable to 0 (using the client) if we should not write files and you will write data using the callbacks
- Call back for start acquisition

- Call back for acquisition finished
- start tcp server thread
- as long as keeprunning is true, usleep for a second
- interrupt caught, delete slsReceiverUsers object and exit
- parent process, increment number of running processes
- Print Ready and Instructions how to exit
- Parent process waits for all child processes to exit by sleeping till numrunning is 0

Definition at line 140 of file mainReceiver.cpp.

3.3.3.4 void sigChildExitedHandler (int *sig*)

Child Exit Signal Interrupt Handler When a child process exits, this function is called, blocks until all child processes exit & decreases the variable indicating number of running processes

Parameters:

sig signal enum

Definition at line 55 of file mainReceiver.cpp.

3.3.3.5 void sigInterruptHandler (int *p*)

Control+C Interrupt Handler Sets the variable keeprunning to false, to let all the processes know to exit properly

Definition at line 65 of file mainReceiver.cpp.

3.3.3.6 int StartAcq (char * *filepath*, char * *filename*, uint64_t *fileindex*, uint32_t *datasize*, void * *p*)

Start Acquisition Call back slsReceiver writes data if file write enabled. Users get data to write using call back if registerCallBackRawDataReady is registered.

Parameters:

filepath file path

filename file name

fileindex file index

datasize data size in bytes

p pointer to object

Returns:

ignored

Definition at line 80 of file mainReceiver.cpp.

3.3.4 Variable Documentation

3.3.4.1 pid_t childPid[NUM_RECEIVERS]

Definition at line 41 of file mainReceiver.cpp.

3.3.4.2 bool keeprunning

Variable is true to continue running, set to false upon interrupt

Definition at line 43 of file mainReceiver.cpp.

3.3.4.3 int numrunning

Variable indicating number of child processes running

Definition at line 45 of file mainReceiver.cpp.

3.4 slsDetectorUsers.h File Reference

```
#include <stdint.h>
#include <string>
```

Classes

- class slsDetectorUsers

  Class for detector functionalitiesto embed the detector controls in the users custom interface e.g. EPICS, Lima etc.

3.5 slsReceiverUsers.h File Reference

```
#include <stdio.h>
#include <stdint.h>
```

Classes

- class slsReceiverUsers

  Class for implementing the SLS data receiver in the users application. Callbacks can be defined for processing and/or saving data.

Index